# Further Development of AI Tool for Extraction of Roadside Hazards from Videolog Data and LiDAR

**NCDOT Project 2023-22**
**FHWA/NC/2023-22**
**May 2025**

Hong Yi, Matthew Satusky, David Borland,
Meghna Chakraborty, Mike Vann, Raghavan Srinivasan,
and Chris Bizon
University of North Carolina, Chapel Hill

RESEARCH & DEVELOPMENT

## Technical Report Documentation Page

| | | | |
|---|---|---|---|
| 1. Report No.<br>**FHWA/NC/2023-22** | 2. Government Accession No. | 3. Recipient's Catalog No. | |
| 4. Title and Subtitle<br>**Further Development of AI Tool for Extraction of Roadside Hazards from Videolog Data and LiDAR** | | 5. Report Date<br>**May 2025**<br>6. Performing Organization Code | |
| 7. Author(s)<br>**Hong Yi[1], Matthew Satusky[1], David Borland[1], Meghna Chakraborty[2], Mike Vann[2], Raghavan Srinivasan[2], and Chris Bizon[1]** | | 8. Performing Organization Report No. | |
| 9. Performing Organization Name and Address<br>**[1]Renaissance Computing Institute, University of North Carolina, Chapel Hill, NC, USA**<br>**[2]Highway Safety Research Center, University of North Carolina, Chapel Hill, NC, USA** | | 10. Work Unit No. (TRAIS)<br>11. Contract or Grant No. | |
| 12. Sponsoring Agency Name and Address<br>**North Carolina Department of Transportation**<br>**Research and Development Unit**<br>**1549 Mail Service Center**<br>**Raleigh, North Carolina 276699-1549** | | 13. Type of Report and Period Covered<br>**Final Report**<br>**Aug 2023 to January 2025**<br>14. Sponsoring Agency Code<br>**RP2023-22** | |
| 15. Supplementary Notes | | | |

16. Abstract

Automated detection and geolocation of roadside objects are critical for effective roadway safety analysis and transportation planning, particularly in rural areas. The goal of this project was to detect and geolocate roadside objects using a videolog comprising over 43 million images of North Carolina's rural roads. This study describes an approach for detecting and geolocating stationary roadside objects by fusing airborne LiDAR data with videolog images. While multi-modal sensor fusion has been widely studied and applied in autonomous navigation for enhanced spatial perception, to the best of our knowledge, existing methods all assume known sensor parameters and dense spatiotemporal resolution to facilitate spatiotemporal data alignment. However, in practice, datasets may have incomplete sensor metadata and sparse spatiotemporal resolution. We aimed to enable automated detection and geolocation of roadside objects using videolog data comprising over 43 million images of North Carolina's rural roads. The videolog lacks camera intrinsic and pose parameters, and due to temporal downsampling of the initial video capture, consecutive images are spaced 26 feet apart, and GPS coordinates must be approximated. To address these limitations, the project team integrated airborne LiDAR data with videolog images through a novel data registration and alignment approach that estimated missing camera parameters through minimization of alignment errors between videolog road lane markings and projected LiDAR Road edges, enabling more accurate computation of object bearings in our geolocation pipeline. The project team was able to apply this approach to detect utility poles in the roadside. This work contributes a practical and scalable solution to the often-overlooked challenge of sensor fusion with incomplete camera metadata.

| | | | |
|---|---|---|---|
| 17. Keywords<br>*AI, Automated Detection, Videolog, Airborne LiDAR* | | | 18. Distribution Statement |
| 19. Security Classif. (of this report)<br>unclassified | 20. Security Classif. (of this page)<br>unclassified | 21. No. of Pages<br>38 | 22. Price |

**Form DOT F 1700.7** (8-72) **Reproduction of completed page authorized**

# DISCLAIMER

The contents of this report reflect the views of the authors and are not necessarily the views of the North Carolina Department of Transportation or the University of North Carolina. The authors are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the North Carolina Department of Transportation at the time of publication. This report does not constitute a standard, specification, or regulation.

# ACKNOWLEDGEMENTS

This research was sponsored by the North Carolina Department of Transportation.

# EXECUTIVE SUMMARY

Automated detection and geolocation of roadside objects are critical for effective roadway safety analysis and transportation planning, particularly in rural areas. The goal of this project was to detect and geolocate roadside objects using a videolog comprising over 43 million images of North Carolina's rural roads. This study describes an approach for detecting and geolocating stationary roadside objects by fusing airborne LiDAR data with videolog images. While multi-modal sensor fusion has been widely studied and applied in autonomous navigation for enhanced spatial perception, to the best of our knowledge, existing methods all assume known sensor parameters and dense spatiotemporal resolution to facilitate spatiotemporal data alignment. However, in practice, datasets may have incomplete sensor metadata and sparse spatiotemporal resolution. We aim to enable automated detection and geolocation of roadside objects using videolog data comprising over 43 million images of North Carolina's rural roads. The videolog lacks camera intrinsic and pose parameters, and due to temporal downsampling of the initial video capture, consecutive images are spaced 26 feet apart, and GPS coordinates must be approximated. To address these limitations, the project team integrated airborne LiDAR data with videolog images through a novel data registration and alignment approach that estimated missing camera parameters through minimization of alignment errors between videolog road lane markings and projected LiDAR Road edges, enabling more accurate computation of object bearings in our geolocation pipeline. The project team was able to apply this approach to detect utility poles in the roadside. This work contributes a practical and scalable solution to the often-overlooked challenge of sensor fusion with incomplete camera metadata.

# TABLE OF CONTENTS

# LIST OF FIGURES

# Chapter 1. Introduction

## 1.1 Background

Crashes involving a roadway or lane departure are associated with a significant number of fatalities each year in the United States. These crashes include head-on collisions with vehicles from the opposing lane, collisions with roadside objects, and rollover crashes. The Federal Highway Administration estimates that more than 50 percent of traffic fatalities in the United States involve a roadway departure.[1] In North Carolina, more than three-fourths of serious injury and fatal lane departure crashes occur in rural areas, and more than 60% of these involve a fixed object.[2] To reduce the severity and frequency of these crashes, transportation agencies require timely and accurate information on roadside objects across extensive rural roadway networks. However, manual inspection and data collection in these environments are labor-intensive, time-consuming, and can pose safety risks to field personnel. Consequently, there is a growing need for automated methods to detect and geolocate roadside objects, i.e., assign geographic coordinates (latitude and longitude) to each object, to enable scalable, data-driven roadway safety analysis and transportation planning, especially in rural areas. Many state agencies have acquired videologs covering portions of their roadway networks. Leveraging these videologs through automation offers the potential to significantly enhance the efficiency and effectiveness of road safety assessments and transportation planning.

In 2018 and 2019, NCDOT collected videolog data for all secondary roads (over 54 thousand miles) in NC, 76% of which are classified as rural roads. The resulting data consists of images sampled every 26 feet using three front-facing cameras; an example image set is shown in Fig. 1. The dataset lacks essential camera intrinsic parameters (e.g., field of view) and extrinsic pose information, such as accurate GPS locations, which presents major challenges for conventional image-based object geolocation techniques.

In prior work [32], this project team established the feasibility of using AI to identify guardrails and utility poles in NCDOT's videolog data. We trained convolutional neural network (CNN) models capable of detecting these roadside objects with high accuracy—90% for guardrails and 88% for utility poles. To facilitate this process, we developed a web-based annotation tool that enabled efficient user labeling of training data through an iterative active learning process to support model development and evaluation as summarized in Fig. 1.

[1] https://highways.dot.gov/safety/RwD

[2] https://connect.ncdot.gov/groups/echs/Documents/2024/2024%20NC%20SHSP.pdf

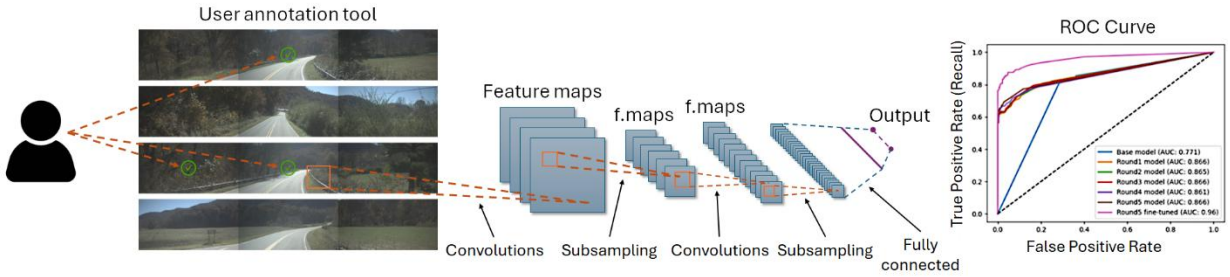


**Fig. 1** Prior work summary plot. (a) A graphical tool enables users to label images as having a particular roadside object such as a guardrail (left). (b) These labels are used to train a convolutional neural network, which predicts whether unlabeled images contain the object of interest (middle). (c) These predictions are validated against human responses, showing good prediction accuracy (right).

Despite the success of the initial feasibility study, this work had two primary limitations. First, while the models could detect the presence of a roadside object in an image, they could not geolocate it. For instance, the models could not distinguish between a pole close to the roadway, which may pose a hazard during lane departures, and a pole that is farther away but still visible in the image. Second, the models lacked the ability to extract terrain or topographic features, such as side slope, which are important in assessing roadside risk.

## 1.2 Research Objective and Scope

To address these limitations and better meet NCDOT's needs, the objective of this project was to extend our earlier work by fusing videolog images with airborne LiDAR data to estimate the spatial location of detected roadside objects and extract detailed geometric and topographic information. Our data fusion approach enables each detected object to be assigned a geographic location based on spatial cues in the LiDAR data. In addition, LiDAR data fusion provides rich geometric context, making it feasible to extract road geometry, fixed object density, clear zones, and side slope characteristics at each identified object location for comprehensive roadside safety analysis.

## 1.3 Research Approach and Contribution

We build our geolocation pipeline based on the pipeline developed by Krylov et al. [12], which automatically detects and computes the GPS coordinates of recurring stationary objects of interest using street view imagery. Their processing pipeline uses a CNN model to detect objects in images, applies monocular depth estimation to estimate the distance of detected objects from the camera, and uses a custom Markov Random Field (MRF) model to perform triangulation for automatic mapping and geolocation of objects in complex scenes. This approach is effective for street view imagery with known camera position and orientation (i.e., bearing towards north) for each image; however, this information is not provided in our videolog data. To overcome this limitation, we have introduced a novel LiDAR-based registration and alignment module to estimate missing camera parameters by minimizing alignment errors between videolog-derived road lane markings and projected LiDAR road edges, enabling more accurate object bearing computation for geolocation using their pipeline.

While multi-modal sensor fusion has been widely studied and applied in domains such as autonomous navigation for enhanced spatial perception, to the best of our knowledge, existing methods all assume known sensor parameters and dense spatiotemporal resolution to facilitate accurate spatiotemporal data alignment. However, in practice, datasets—such as the videolog

from NCDOT—may contain incomplete sensor metadata and sparse spatiotemporal resolution. By using utility poles as a case study, we demonstrate that our extended pipeline can effectively detect and geolocate roadside objects under real-world constraints. Our work thus offers a practical and scalable solution to the often-overlooked challenge of sensor fusion with incomplete camera metadata and limited spatiotemporal resolution.

The specific contributions of our work include:

- A scalable, data fusion-based geolocation pipeline for roadside objects that addresses the often-overlooked challenge of sensor fusion with incomplete camera metadata.
- An extension of Krylov et al. [12]'s object geolocation pipeline by incorporating additional sensor fusion-related components. For those components introduced in Krylov et al.'s pipeline—such as image segmentation, monocular depth prediction, and MRF-based geotagging—we used alternative models or extended their methods to better suit our application, offering new insights into geolocation challenges.
- A novel data registration and alignment method that integrates airborne LiDAR and videolog imagery. This method estimates missing camera parameters by minimizing alignment errors between videolog road lane markings and projected LiDAR road edges, enabling more accurate object bearing computation in the geolocation pipeline.

## 1.4 Report Organization

The remainder of the report is organized as follows. Chapter 2 is a summary of previous related research. Chapter 3 describes our data fusion-based geolocation pipeline and methodology. Chapter 4 presents our validation results and analysis. Finally, Chapter 5 provides the conclusions and future directions.

# Chapter 2. Review of Relevant Prior Work

In this section, we present related work on image- and LiDAR-based object detection, classification, and geolocation, along with sensor fusion approaches.

## 2.1 Image-Based Object Detection, Classification, and Geolocation

In recent years, machine learning techniques, particularly deep learning (DL) convolutional neural networks (CNNs), have shown exemplary performance in automated object detection and classification from images and videos [11]. However, object geolocation must handle additional complexity by recognizing objects appearing in multiple images and merging them into one single geolocation using re-identification-, tracker-, or triangulation-based methods [27].

In re-identification-based methods, a model performs object detections using multiple image frames and outputs a single prediction for an object from the multiple input frames. Re-identification-based methods were proposed by Nassar et al. [18, 19], but their models require determining a fixed number of input image frames for detecting objects before training, which is impractical for real-world situations [27].

In tracker-based methods, objects between frames are associated and tracked in a model to compute a final prediction. For example, Chaabane et al. [3] constructed a CNN consisting of an object pose regression network and an object matching network, which used the camera's intrinsic matrix along with six different image perspectives collected by six cameras. Wilson et al. [27] proposed a multi-class tracking-based deep learning approach for geo-localization of objects in multiple classes from images captured by a single camera, requiring images' GPS coordinates and headings. These approaches are not applicable to our use case, as our videolog images lack critical camera metadata, including intrinsic parameters, accurate GPS coordinates, and heading information.

Triangulation-based approaches use a classic triangulation method to compute an object's geolocation using the depth to an object in an image and the image's GPS coordinates and headings, followed by a final clustering algorithm to cluster repeated object occurrences into one single geolocation. The first triangulation-based approach for object geolocation from street view imagery was presented by Krylov et al. [12], who proposed a custom Markov Random Field (MRF) model to perform object triangulation for geolocation after segmenting objects in the images using a CNN-based semantic segmentation model and estimating object distance from the camera using a monocular depth estimation model. Their MRF model requires images' GPS coordinates and headings (i.e., bearings towards north) to perform triangulation. Similarly, Zhang et al. [34] applied DL to identify utility poles with crossarms in Google Street View images and estimate their spatial positions with a line-of-bearing (LOB)-based triangulation method. We built our object geolocation pipeline upon Krylov et al.'s [12] triangulation-based object geolocation approach.

## 2.2 LiDAR-Based Object Detection and Classification

While street view images and videos are widely used for object detection and geolocation due to the rich visual details they provide, cameras are sensitive to varying lighting and weather conditions, and suffer from imprecise geolocation and limited depth information. To overcome

these limitations, DL-based LiDAR 3D point cloud pointwise classification and semantic segmentation (e.g., SqueezeSeg [28], CENet[4]) has drawn increasing attention for accurate, real-time, and robust environment perception and understanding, especially for autonomous driving. Sun et al. [23] presented a toolbox to support the exploration, comparison, and benchmarking of convolutional LiDAR segmentation models. Li et al. [14] gave a comprehensive survey of DL for LiDAR point clouds in autonomous driving, summarizing existing LiDAR point cloud datasets for model training, validation, and benchmarking, general 3D DL frameworks, and remaining challenges. Alaba et al. [1] presented a survey on DL-based LiDAR 3D object detection and feature extraction techniques for autonomous driving, including a summary of the commonly used LiDAR 3D coordinate systems and encoding techniques.

## 2.3 Sensor Fusion Approaches

While DL models based on vision or LiDAR data have demonstrated strong performance in object classification and segmentation tasks, their effectiveness in precise object geolocation and 3D spatial understanding remains limited when used in isolation. In contrast, multi-modal sensor fusion systems have shown substantial potential in enhancing spatial perception by leveraging the complementary strengths of different types of sensor modalities through appropriate fusion strategies [25]. By integrating information from both visual and LiDAR data sources, these systems can overcome the individual shortcomings of each modality, such as inaccurate monocular depth estimation from images or sparse object representations in LiDAR scans. However, the performance of multi-modal fusion approaches is often hindered by challenges such as spatiotemporal misalignment between sensors, domain discrepancies, and varying levels of noise across data sources. To fully realize the benefits of multi-modal perception, it is essential to develop more robust spatiotemporal registration techniques and advanced data fusion strategies that can effectively reconcile differences across sensor domains and enhance the overall perception performance [24].

While sensor fusion between LiDAR and other modalities, such as camera images, has become an important area of research, DL-based fusion methods still face notable challenges. These approaches must balance accuracy with algorithm complexity due to data redundancy, and there is still a huge gap between algorithm design and practical applications in the real world [29]. Wang et al. [26] conducted a comprehensive review of recent DL-based multi-modal 3D detection networks, particularly focusing on LiDAR-camera fusion. Their analysis centers on three key dimensions of fusion design: when to fuse (fusion stage), what to fuse (fusion inputs), and how to fuse (fusion granularity). These design decisions critically influence system performance and typically involve projecting LiDAR points into the image plane using homogeneous transformations to establish a 3D-2D correspondence between the two modalities [26].

A growing body of research has focused on developing novel data fusion architectures for effective alignment of LiDAR and image data. For example, Huang et al. [9] introduced EPNet, a learning-based fusion framework for 3D object detection that combines LiDAR point features with semantic image features through a LiDAR-guided Image Fusion (LI-Fusion) module. This module performs point-wise projection of LiDAR data onto the image plane to establish the correspondence between LiDAR and image data and adaptively weights the importance of the image semantic features, effectively enhancing relevant image features while suppressing noisy

or interfering image features. Similarly, Li et al. [15] explored fusion strategies for improving multimodal 3D object detection by addressing feature alignment challenges. They proposed two techniques, InverseAug, which projects 3D key points—derived after the data augmentation of the original LiDAR point cloud during training—to 2D camera features using the LiDAR and camera parameters, and LearnableAlign, which leverages cross-attention to dynamically learn the correlation between a LiDAR feature and its corresponding camera features.

Alignment of multi-sensor data in fusion-based approaches typically relies on known LiDAR and camera parameters to project a 3D LiDAR point cloud onto the 2D image plane. Similarly, vision-based object geolocation methods using street view imagery often require not only image GPS coordinates, but also camera intrinsic and pose parameters, such as headings or bearing information, to estimate object geolocations effectively. However, in real-world settings, datasets may contain incomplete sensor metadata. For example, the videolog imagery collected by NCDOT includes approximate GPS coordinates with consecutive images spaced 26 feet apart but lacks critical camera intrinsic and pose parameters. In addition, camera parameters of the Mapillary Vistas dataset [20], composed of data pulled from heterogeneous sources, are not readily accessible. To address this limitation, we developed a novel data registration and alignment approach that fuses airborne LiDAR data with videolog images. Our approach estimates the missing camera parameters by minimizing alignment errors between visible road lane markings in the videolog imagery and the projected road edges derived from LiDAR, thereby enabling accurate computation of object bearings in our geolocation pipeline. We describe our geolocation pipeline and methodology in the following section using utility pole detection and geolocation as a case study.

# Chapter 3. Methodology

To enable accurate geolocation of roadside utility poles using videolog images with incomplete camera metadata, we have adapted and extended the geolocation pipeline introduced by Krylov et al. [12]. While their approach leverages street view imagery with known camera positions and orientations (i.e., bearing towards north), our work addresses the practical challenges of working with videolog imagery that lacks both intrinsic and extrinsic camera parameters. Specifically, we introduce a novel LiDAR-based registration and optimization module that estimates camera orientation by aligning road lane markings detected in videolog images with projected road edge boundaries extracted from airborne LiDAR data. This alignment facilitates more accurate object bearing estimation, which is critical for the MRF-based triangulation and geolocation approach used by Krylov et al.

Fig. 2 illustrates the overall structure of our geolocation pipeline, including the major components and data flows. The two key components—camera parameter estimation via road alignment and mapping input computation—are highlighted in yellow. These components transform raw data inputs (i.e., videolog images and airborne LiDAR data) into the spatial and geometric inputs required for the final MRF-based geolocation.

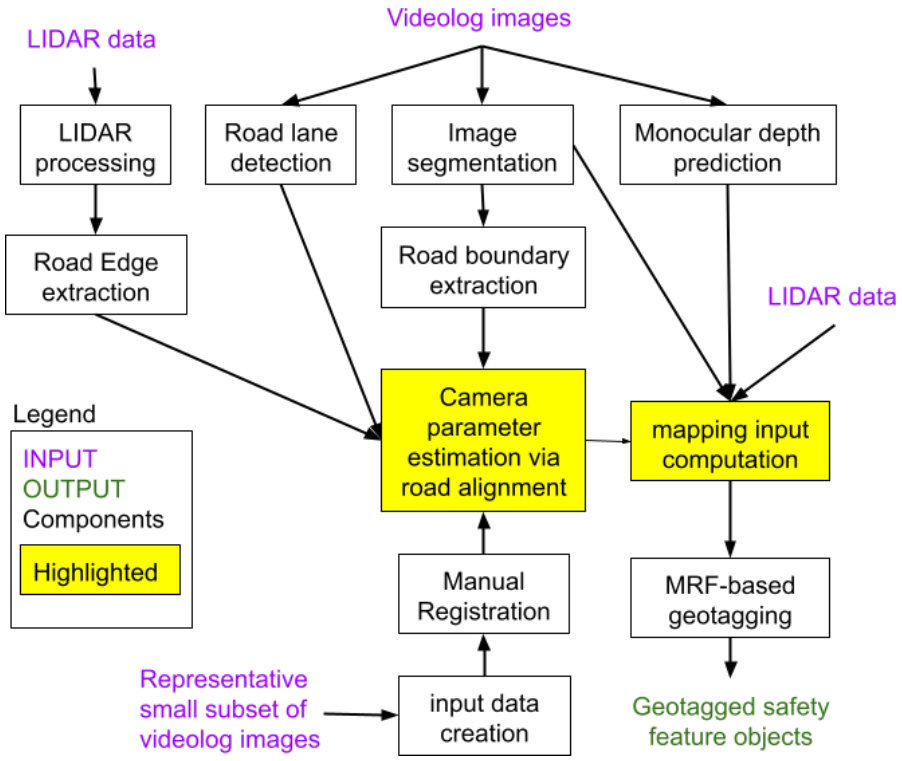


**Fig. 2** Pipeline components for roadside object geolocation.

In the following sections, we describe each component of the pipeline using utility poles detection and geolocation as a case study. Our geolocation pipeline is designed to be generalizable, enabling geolocation of not only utility poles but also other roadside safety objects of interest, such as trees, buildings, and guardrails.

### 3.1 Image Segmentation

Krylov et al. [12] employed a semantic segmentation model based on the Fully Convolutional Neural Network (FCNN) architecture [22] to detect objects of interest. The model outputs pixel-level labels that can be directly used as masks in the depth estimation step of their geolocation pipeline. However, the segmentation model used in their implementation is not publicly available.

To select an effective semantic segmentation model for our geolocation pipeline, we qualitatively evaluated two publicly available deep learning-based image segmentation frameworks on a mountainous route in our videolog: MIT SemSeg [35, 36] and OneFormer[10]. Models in the SemSeg library were trained on the ADE20k scene parsing dataset, while the subset of OneFormer models evaluated were trained on either the Cityscapes [5] or Mapillary Vistas [20] dataset.

For each framework, multiple top-performing model architectures were tested on raw videolog images, as well as a series of increasingly downsampled versions to determine a minimum resolution for acceptable performance. Broadly, the OneFormer models outperformed similarly-sized SemSeg models for detecting utility poles, with larger models generally providing the best segmentation results. The largest OneFormer model, ConvNeXt-XL, is available pretrained on Cityscapes or Mapillary Vistas data. On our images, the Mapillary Vistas version performed marginally better, and we therefore selected that model moving forward. We observed a marked decrease in utility pole detection when images were scaled below 640 X 512 pixels (width X height), so that resolution was chosen as a balance between speed and accuracy.

One drawback of the larger segmentation models is that they often misidentify extraneous pole-like objects (e.g., fence and sign posts) as utility poles, resulting in a high number of false positives (FPs), as shown in Figure 3. To reduce FPs in pole segmentation, we developed a rule-based post-processing method. This method begins by removing wire extensions from segmented poles using morphological operations, specifically, erosion followed by dilation. It then filters out FPs using empirically derived rules based on object height-to-width aspect ratio and the relationships between object height and estimated object depth. While we can remove much of this pole-like noise through post-processing, there remain FPs that closely resemble true utility poles and are difficult to distinguish based on visual appearance alone. Further improvements in segmentation accuracy could be achieved through the use of more advanced segmentation models or more comprehensive post-processing algorithms that better balance false positives and false negatives.

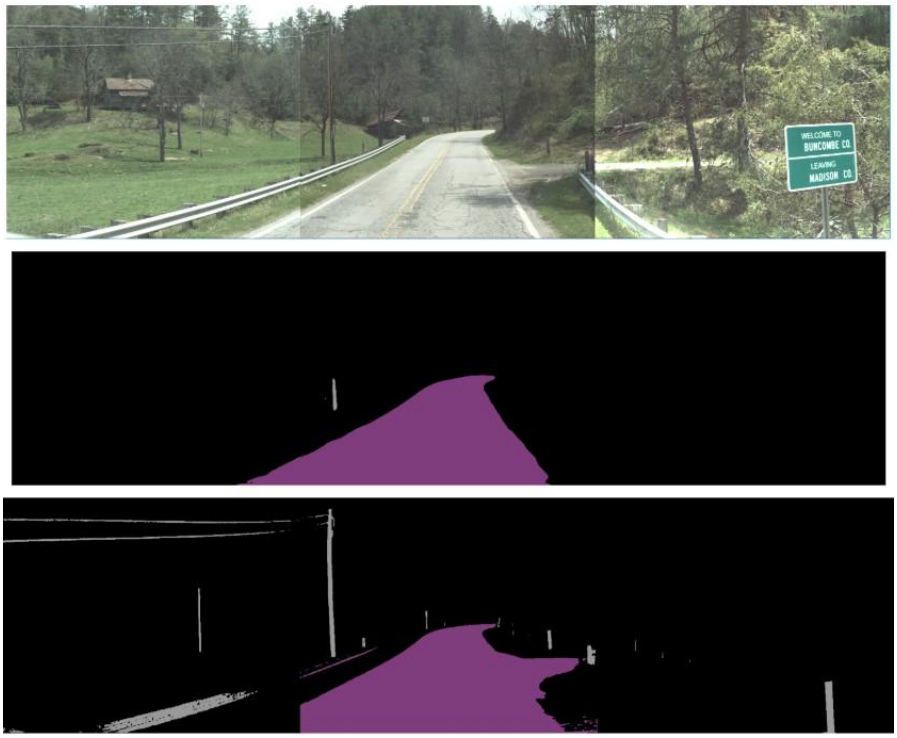


**Fig. 3** Comparison of segmentation results of an example image (top) from the MIT model and the OneFormer model. The MIT model segmentation (middle) detected a much shortened pole with no connected wires, while the OneFormer model segmentation (bottom) detected the accurate pole with connected wires, along with other pole- or wire-like objects

## 3.2 Camera Parameter Estimation via Road Alignment Optimization

We developed an optimization module to estimate unknown camera parameters for each videolog image by aligning road edges extracted from airborne LIDAR data with road lane markings detected in the images. This optimization process requires a set of initial camera parameters to serve as a baseline for minimizing alignment errors. To generate these initial parameters, we built a manual registration tool that enables users to calibrate a representative image for each unique image resolution. The resulting baseline camera parameters are then applied to all images sharing that resolution for optimization by minimizing road alignment errors. In the following subsections, we provide a detailed description of the optimization approach, including the extraction of road edges from both LiDAR and image data, as well as the supporting manual registration process.

### 3.2.1 Road Edge Extraction from LiDAR Data

Each airborne LiDAR point provided by the NCDOT contains the world coordinate, elevation, and a classification (e.g., ground, road, building) (Figure 4a). We performed voxel rasterization on the raw LiDAR point cloud to reduce the volume of data and create a uniform distribution of data points. The world coordinates are divided into a 1 ft. x 1ft. X 1ft. voxel grid, and a data point is placed at the grid coordinate of any voxel containing at least one LiDAR point. If any points within that voxel are classified as road or bridge, the rasterized point assumes that label to prevent gaps in the roadway; otherwise, it is categorized as the most frequently occurring classification within that voxel (Figure 4b). To find the roadway edges, we filter the rasterized LiDAR data for the “highest hit” (the point with the highest elevation) in each world X-Y

coordinate (Figure 4c) and create a 2D overhead image of the highest hit classes (Figure 4d). We then convert the classes to a binary image, designating either roadway (1) or other (0), and perform binary dilation and erosion to identify the edge pixels. Finally, the coordinates for edge pixels in the binary image are converted back to world coordinates.

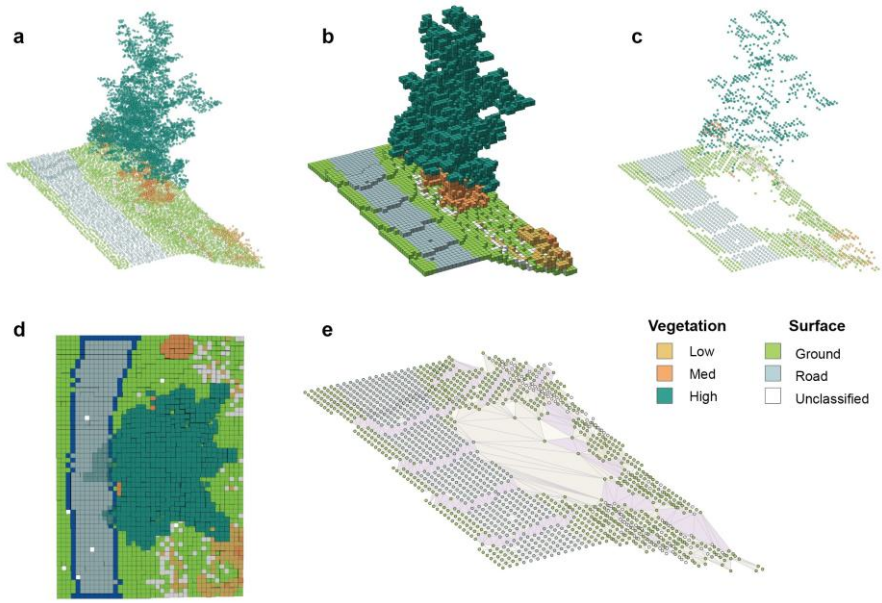


**Fig. 4** LiDAR processing pipeline. **a)** A 3D visualization of LiDAR world coordinate points for a scene containing a roadside tree. **b)** The scene is rasterized by dividing into a voxel grid and assigning any occupied voxel the most frequently occurring class or roadway, if any road points fall within. **c)** Rasterized points are reduced to the highest occupied voxel for each X-Y coordinate plus any roadway voxels. **d)** A 2D overhead image is created from the reduced, rasterized point cloud, and the roadway edges are identified (dark blue). **e)** The terrain surface is reconstructed using a Delaunay triangulation of the ground and road points for determining visibility through ray casting.

### 3.2.2 Road Edge Extraction from Videolog Images

Extracting road edge features from videolog images is a critical first step in our camera parameter optimization process, as these features are used to align with LiDAR extracted road edges. Through experiments using our manual registration tool, we observed that aligning LiDAR road edges with painted lane markings—rather than with road boundaries segmented from the images—resulted in more accurate registration. To extract these lane markings from the videolog images, we employed the image segmentation model SegFormer [30]. The base model, pretrained on the Cityscapes dataset, was equipped with a binary classification head and fine-tuned using a 10,000-image subset of lane marker labels from the BDD100k dataset [33]. Model tuning was performed on a single NVIDIA RTX 3090 GPU for 30 epochs.

In our alignment process, we filtered out the detected middle lane markings and used only the left and right lane markings to correspond to the road edges captured in the LiDAR data. Specifically, we removed middle lane markings by estimating a middle lane axis using clustered centroids from rows with dense point density, then filtering out points near this axis based on their perpendicular distance to the axis. One advantage of using lane markings for alignment is their relative robustness to noise. Unlike segmented road boundaries, which can include irrelevant features such as parking lots, driveways, or shoulders adjacent to the main road, lane markings tend to represent road features for alignment more consistently without including this

extraneous noise. However, lane-based alignment may fail to capture intersecting roads if their lane markings are missing or poorly detected, as shown in Figure 5. In addition, the lane detector often misses distant road segments, which can be particularly important for accurate alignment in scenes with curved roads or long stretches of roadway.

**Fig. 5** An example image with lane markings segmented by our fine-tuned SegFormer model, overlaid on the original image

To overcome the limitations of relying solely on lane markings for alignment— particularly in complex road environments such as intersections or distant curved road segments—we incorporated road boundaries extracted from segmented road pixels using the OneFormer model. For instance, in the example shown on the left of Figure 5, an intersecting road on the right lacked visible lane markings, causing the lane detection model to miss it entirely. By integrating road boundary segmentation with lane detection, we are able to recover such missed segments and incorporate them into the alignment process, improving road alignment in challenging road scenarios.

However, this integration can also introduce unwanted artifacts, such as parking lots, driveways, and occlusions from road-blocking vehicles. To mitigate these issues, we applied the detected left and right lane markings as spatial masks and retained only the road boundary pixels located outside the region enclosed by the lanes. As illustrated in Figure 6, this integrated filtering approach effectively removed noisy segments—such as occlusions from vehicles—while preserving important features like intersecting road boundaries for alignment. Nonetheless, unwanted artifacts may still be present in the processed data. As such, the alignment optimization algorithm must be robust to occasional segmentation errors or noise introduced through this integration process.

**Fig. 6** Filtered and integrated road edge segmentation result (right) of an example image (left), where the filtered and integrated road edge segmentation pixels are shown in light blue, and projected LiDAR road edge pixels are shown in dark blue.

### 3.2.3 Manual Registration

The manual registration step in our pipeline utilizes a standalone interactive tool (Figure 7) that we developed to calibrate a representative image for each unique image resolution. The tool is browser-based and uses the Three.js library to enable the user to interactively adjust the projection—3D position, rotation, and field of view—of input 3D LiDAR points onto a 2D image and its resulting road edge segmentation. This calibration process produces a set of initial baseline camera parameters, which are then used as the starting point for all images with the same resolution. These baseline camera parameters serve as input to the subsequent optimization module, which refines the camera parameters by minimizing road alignment errors.

We also leveraged the manual registration tool to support parameter exploration and to debug the automatic registration pipeline. For example, we used it to examine the relationships between camera parameters for front-view images and their corresponding left and right side-view images. Through the interactive examination, we found that the camera parameters derived from aligning projected LiDAR road edges with detected road edges in front-view images were also effective for the side-view images. This was evident in the strong visual alignment between LiDAR points corresponding to buildings and houses and their counterparts in the side-view images. As such, we applied the optimized camera parameters obtained from the front-view images to the stitched left+front+right view images, enabling the inclusion of side-view images for more accurate object geolocation, as roadside objects of interest frequently appear in side-view images.

### 3.2.4 Road Alignment-Based Optimization

Our alignment pipeline estimates the camera pose parameters for each image through an optimization algorithm that minimizes alignment errors between projected LIDAR road edges and the segmented road edges in the videolog images, starting from the baseline parameters obtained using the manual registration tool. To achieve this alignment between 2D road edge pixels in the image coordinate system and 3D LIDAR road edge vertices, we implemented a coordinate transformation pipeline using righthanded coordinate systems consistent with OpenGL conventions (https://learnopengl. com/Getting-started/Coordinate-Systems).

**Fig. 7** Manual registration tool screenshot.

Specifically, we first transform the LiDAR road edge points from their original geographic coordinate reference system (CRS) (NAD83, EPSG:6543) into a right-handed 3D world coordinate system centered at the camera. In this world coordinate system, the camera is positioned at the origin, the negative z-axis points in the camera's bearing direction (from the current to the next image camera location), the positive y-axis points upward (aligned with the LiDAR elevation axis but orthogonalized to be perpendicular to the z-axis), and the positive x-axis points to the right, perpendicular to both the y- and z-axes.

Figure 8 shows a diagram of the transformation steps. First, each camera's location in the LiDAR CRS is obtained by transforming its latitude-longitude pair (originally in WGS84 CRS) into X-Y coordinates in the LiDAR's NAD83 CRS, while the camera height (Z) is approximated by nearest-neighbor interpolation of surrounding LiDAR points. Using the computed camera position and bearing direction, we define the world coordinate frame for each image. Each LiDAR road edge point (restricted to the camera's estimated field of view (FOV) and a distance threshold from the camera) is then transformed into this world coordinate frame. These transformed LiDAR points are subsequently projected onto the camera's image plane through a perspective projection and finally mapped to pixel coordinates in the 2D screen coordinate system for alignment with the segmented road edges.

We employed the Nelder-Mead simplex optimization algorithm [6] to iteratively minimize a grid-based pairwise distance function between the projected LiDAR road edges and the segmented road boundaries in the screen coordinate system, estimating the optimal camera parameters with minimal alignment error from the initial baseline parameters. The grid-based distance function constrains the search space by limiting segmented road boundary pixels to a rectangular region centered around each projected LiDAR point pixel, reducing the likelihood of mismatches. To further improve alignment accuracy, particularly in complex scenes such as

narrow or remote roads, we classified both LiDAR road edge points and segmented road boundary pixels into left and right roadside based on the camera's bearing direction, and computed pairwise distances only between point pairs on the same side. In addition, to improve robustness against noisy or poor-quality input data, we incorporated empirically derived range constraints for each camera pose parameter into the optimizer, preventing the generation of unrealistic camera parameters due to erroneous road edge detection inputs.

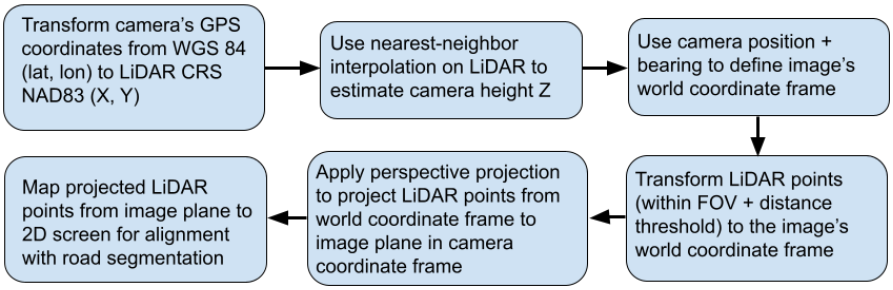


**Fig. 8** Diagram of LiDAR data transformation steps.

To improve road edge fitting and object geolocation, we integrated a ray casting algorithm to remove LiDAR points obscured from the camera by the surrounding terrain. LiDAR points labeled as ground or road are selected, and a surface mesh is created using a Delaunay triangulation of the point world coordinates (Figure 4e). To determine the visibility of a LiDAR point, a vector ray is cast from the camera position to the point. If the ray intersects any surface triangle, the point is occluded and removed from the scene. The LiDAR points are organized by projected screen coordinates in a quadtree structure, and screen blocks are processed in parallel to reduce computational overhead. To further manage computational cost, the occlusion filtering was applied once for each image upon the initial scene reconstruction prior to sending the LIDAR points to the optimizer for iterative camera parameter refinement.

## 3.3 Mapping Input Computation

Similar to the geolocation pipeline proposed by Krylov et al. [12], our pipeline requires as inputs the camera latitude and longitude, as well as the estimated absolute monocular depth and bearing for each detected utility pole in an image to perform MRF-based triangulation for object geolocation (refer to Figure 3 in [12] for the original triangulation diagram). Since the camera latitude and longitude are already available for each image, the following subsections focus on our methods for estimating the absolute monocular depth and bearing for detected poles.

### 3.3.1 Monocular Depth Estimation and Absolute Metric Mapping

Krylov et al. [12] employed the FCNN-based depth estimation pipeline introduced by Laina et al. [13]. This pipeline is composed of a fully convolutional ResNet-50 backbone [8], followed by a cascade of residual up-projection blocks, to generate a dense absolute monocular depth map at the native image resolution. However, the download link for this FCNN depth estimation model is no longer active, preventing us from directly testing it for absolute metric depth estimation. Moreover, most published monocular depth estimation models predict only relative pixel depths up to an unknown scale, and converting these relative predictions to absolute metric depths typically requires per-image scaling factors derived from ground truth measurements [2]. By

"metric depth", we refer to the true physical distance from the camera to the corresponding point in the 3D world for each pixel.

Since our geolocation algorithm depends on sufficiently accurate estimates of the metric depth for each segmented utility pole, the precision of these depth inputs directly affects the geolocation outcome. To improve geolocation accuracy, we investigated multiple depth prediction models and conducted extensive research into mapping predicted relative monocular depths to absolute metric depths.

First, we evaluated the MiDaS model [21] in combination with the calibration method proposed by McCraith et al. [17] to convert relative depth estimates to metric depths. However, our testing revealed that the resulting converted metric depths were not sufficiently accurate when compared against LiDAR ground truth measurements. More recently, ZoeDepth [2] introduced the first monocular depth prediction framework capable of maintaining metric scale across diverse domains. We validated the publicly available pre-trained ZoeDepth models against LiDAR measurement and found that, while ZoeDepth provided consistent scaling, its predicted depths systematically underestimated actual distances. As a result, ZoeDepth's output was not directly applicable to our geolocation pipeline.

More recently, Yang et al. introduced the Depth-Anything model [31], providing the most capable monocular depth estimation framework to date. Their evaluations demonstrated that Depth-Anything achieves higher zero-shot relative depth accuracy than MiDaS and higher zero-shot metric depth accuracy than ZoeDepth. We tested both the relative and metric depth prediction capabilities of Depth-Anything on our data, obtaining validation results consistent with those reported in [31]. However, our testing showed that even the Depth-Anything metric depth predictions tended to systematically underestimate distances relative to our LiDAR ground truth. As a result, we adopted Depth-Anything in our pipeline for predicting relative depth maps, while developing a custom mapping procedure to convert these predictions into absolute metric depths.

Due to the inherent ambiguity of monocular vision, monocular depth estimation models can predict relative depth maps accurately up to an unknown global scaling factor, which is typically derived from ground truth measurements such as LiDAR [17, 7]. McCraith et al. [17] proposed a monocular depth calibration approach that infers a per-frame depth scaling factor based on known camera height and a parametric plane fit to the raw depth map of segmented road pixels. Masoumian et al. [16] proposed an alternative approach, modeling the mapping from predicted relative depth to absolute metric depth as a quadratic function involving four coefficients, including camera height and three others estimated via least-squares optimization. We tested both approaches on our dataset, using estimated camera heights. Our analysis revealed that the relationship between predicted relative monocular depths and LiDAR-based metric distances is neither globally linear nor quadratic across an image. Therefore, applying a single global scaling factor per frame was insufficient for our geolocation needs.

After extensive experimentation, we developed a new mapping algorithm based on the perspective projection model. Integrating this algorithm into our geolocation pipeline produced sufficiently accurate absolute depth estimation input for the MRF-based geolocation algorithm. Specifically, our approach leverages the mathematical relationship between 3D world

coordinates and 2D image coordinates in perspective projection. In standard OpenGL-stype perspective projection, the $z$ value in the camera coordinate system is normalized to the $(-1,1)$ to simulate depth compression, where objects farther from the camera appear smaller. The normalized depth $d$ is related to the camera-space depth $z$ through the equation $d = c1/(-z) + c2$, where constants $c_1$ and $c_2$ are derived from the near and far plane distances. Using this mapping, we normalized LiDAR point metric depths $z$ (after applying rotation and translation along the camera bearing direction) to the $(-1,1)$ range consistent with relative monocular depth prediction outputs. Empirically, we observed a strong linear correlation between the predicted monocular depths and these normalized LiDAR depths. Thus, we performed a linear regression fit to establish a mapping function, and then inverted the mapping equation ($z = c1/(c2-d)$) to compute absolute metric depths $z$ for detected objects from their predicted relative monocular depth $d$. A scatter plot comparing computed absolute depths for a segmented utility pole across six consecutive test images against its LiDAR-measured distances is shown in Figure 9. These results confirm that our approach achieves sufficiently accurate absolute metric depth estimates for geolocating utility poles.

### 3.3.2 Bearing Estimation via LiDAR-Image Data Fusion

To compute the bearing angles for segmented utility poles, we developed a data fusion approach that combines LiDAR and image information. First, we rasterized the raw LiDAR point cloud into a 1x1 foot grid and projected the resulting dense LiDAR points onto the image plane using the estimated camera parameters for each frame. For each segmented utility pole, we identified a corresponding LiDAR point located along the approximate line of sight between the camera and the pole. Specifically, we selected the LiDAR point whose projected 2D location was closest to the bottom of the segmented pole in the image. The bearing between the camera and the geographic location of this LiDAR point was then computed and used as the bearing input for the geolocation pipeline.

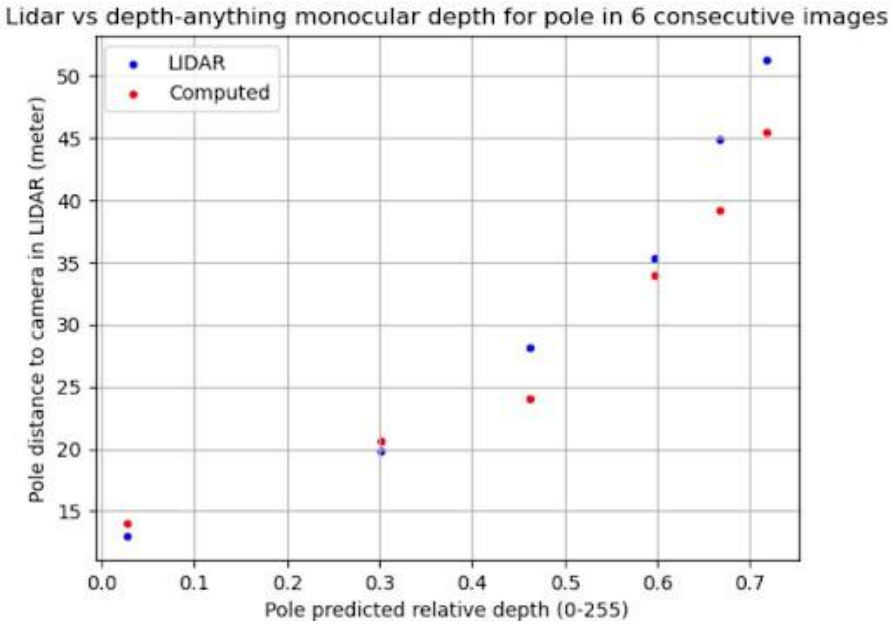


**Fig. 9** Scatter plot of computed metric depths of a segmented pole in six consecutive test images (red) compared with LiDAR-measured distances to the camera along the viewing direction (blue).

## 3.4 Object Triangulation and Geolocation

We adapted the MRF-based object triangulation and optimization approach proposed by Krylov et al. [12] to better meet our geolocation requirements. Their MRF-based method leveraged recurring stationary objects detected in consecutive frames and performed triangulation based on pairwise intersections of view-rays from camera positions. Specifically, the MRF space was defined as all pairwise intersections of view-rays, constrained by a 25-meter maximum camera-to-intersection distance to ensure accurate geolocation near the cameras. Each intersection point was assigned a binary label indicating the presence or absence of an object, and the overall MRF energy was composed of three terms: a unary term enforcing consistency with depth estimation, a pairwise term penalizing segmentation noise and occlusion errors, and a term penalizing view-rays without positive intersections to resist segmentation false positives or objects discovered from a single camera position. The final MRF energy was optimized using a random node-revisiting schedule until a local minimum was achieved or no further changes were accepted. After optimization, hierarchical clustering with a 1-meter maximum intra-cluster distance was applied to produce the final geolocated object locations.

The MRF algorithm requires as inputs the camera geolocations (latitude and longitude) and predicted depth and computed bearing for each detected object in a sequence of images. Bearing inputs directly affect intersection computation and thus strongly influence geolocation accuracy, while depth inputs contribute indirectly through the energy terms, helping validate or reject candidate intersections. Through experiments with synthetic datasets containing ground truth object locations, we verified that the algorithm is highly sensitive to bearing inputs but relatively tolerant of approximate depth inputs.

However, the original MRF approach incorporated randomness in pairwise intersection node selections, leading to slightly different results across repeated runs. In practice, especially when input bearings were not sufficiently accurate, random selection of image pairs may cause

problems when separate detected poles were incorrectly merged into a single geolocated pole or when multiple detections of the same physical pole were not properly merged. Since our estimated camera parameters—and thus computed bearings—are approximate, randomness was undesirable for our application.

To address these issues, we made two modifications. First, we removed randomness from the MRF iterative optimization and considered all image pair intersections, yielding deterministic geolocated outputs independent of multiple runs. Second, to handle cases where multiple detections in close proximity corresponded to the same real-world pole, we applied an additional clustering rule tailored to our sequential imaging setup. Specifically, two detected poles were merged if their intersecting images were captured within 100 feet of each other, the estimated depths decreased consistently as the vehicle approached, and the bearings consistently increased or decreased. Testing on multiple routes confirmed that this rule successfully clustered multiple detections of the same utility pole.

## 3.5 Pipeline Component Summary

To highlight the contributions of each key component in our extended pipeline, we summarize their roles and qualitative impacts (Table 1) based on our iterative development evaluations. These components were retained only if they proved essential for robust performance, as removing them would either break the pipeline or increase geolocation errors in specific scenarios.

**Table 1** Summary of key pipeline components, their roles, and qualitative impacts on performance.

| Component | Role | Qualitative Impact on Performance |
|---|---|---|
| LiDAR-guided bearing selection | Compute object bearings by fusing LiDAR points with segmented images, using the closest projected LiDAR point to the segmented object's base. | Essential for triangulation-based object geolocation; replacing it with image-only bearing computation consistently decreases geolocation accuracy. |
| Road-edge alignment | Estimate camera parameters by minimizing alignment errors between videolog road lane markings and projected LiDAR road edges. | Optional when camera metadata are available, but critical for data fusion and geolocation when using images with unknown camera metadata, as is the case with our videolog. |
| Occlusion filtering via ray casting | Remove obscured LiDAR points using terrain surface reconstruction and ray tracing from the camera. | Optional in flat terrains with minimal occlusions, but essential in complex scenes (e.g., downhill segments), where filtering out occluded LiDAR road edges consistently improves alignment with videolog imagery road edges. |
| Side-view stitching | Combine left, front, and right videolog views into a stitched image for broader field of view. | Essential for comprehensive detection of roadside objects visible only in side views. |
| Deterministic modification of the MRF | Eliminate randomness in MRF optimization by considering all pairwise intersections and add a clustering rule for sequential images for more accurate object geolocation. | Essential to ensure consistent geolocation results across runs with more accurate clustering and prevent potential geolocation errors resulting from random view-ray selections. |

# Chapter 4. Results and Discussion

We tested our geolocation pipeline along a challenging route located in the mountainous western region of North Carolina, characterized by frequent elevation changes, sharp turns, occluded road lanes, and complex intersections. These complexities made it difficult to accurately segment road edge boundaries from images for alignment with LIDAR data and thus presented challenges for accurate camera parameter estimation.

Running our pipeline on this test route resulted in the geolocation of 180 utility poles. Figure 10 shows these 180 geolocated pole locations on a map of this region. From these 180 geolocated pole locations, we randomly sampled 30 for manual validation. ArcGIS Pro version 3.0.0 was used as the GIS software within which a 3D ArcGIS Scene was created to interact with the data. The latitude and longitude (i.e., X and Y coordinates) information of these geolocated locations were the key attributes that were validated manually with ArcGIS Pro. NCDOT provided point cloud LiDAR land classification data from the 2016 Geiger mode collection at 30 pulses per square meter (ppsm), which was clipped to a 100-foot buffer around the test route. Ortho images downloaded from NC One Map were used as a basemap to identify the real-world locations of the poles.

The geospatial validation process involved first creating a LIDAR Aerial Survey (LAS) dataset to reference the LiDAR data. A LAS dataset references one or more .las files, a binary format for storing LiDAR data. This dataset was projected into the NAD 1983 (2011 StatePlane North Carolina FIPS 3200 (US Feet)) as the XY coordinate system and NAVD88 height (ftUS) as the Z (height) coordinate system. A CSV file with the coordinates data of the 30 sampled utility pole locations was then converted into a GIS shapefile and projected into the same XY coordinate system as the LAS dataset. The LiDAR data contained nine land cover classifications; we reviewed them to determine which of these classifications provided the best representation of the poles by overlaying the LAS dataset with the ortho images in a 2D map linked to the ArcGIS Scene. The high vegetation classification provided the best representation of the poles, as it also captured wires, with the medium vegetation classification providing additional context at some sampled locations. Subsequently, these two classifications were filtered from the LAS dataset using the “Make LAS Dataset Layer” geoprocessing tool and displayed separately in the ArcGIS Scene. Next, we identified the point of each LiDAR-represented utility pole that was the closest to the corresponding geolocated pole location from the sampled pole shapefile (Figure 11). The latitude, longitude, and elevation of this LiDAR point were then recorded as the validated location corresponding to the geolocated pole location.

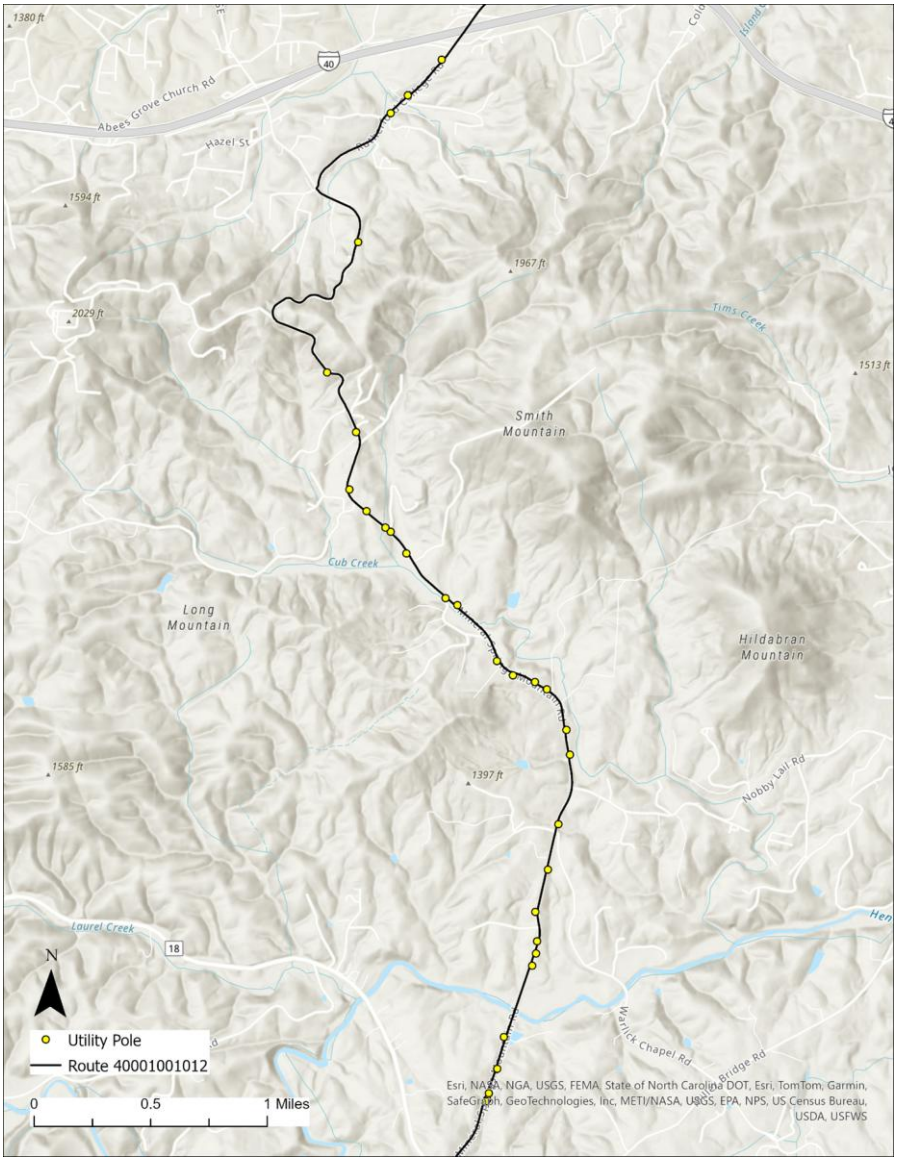


**Fig. 10** A map of the test route with geolocated utility poles indicated as yellow dots.

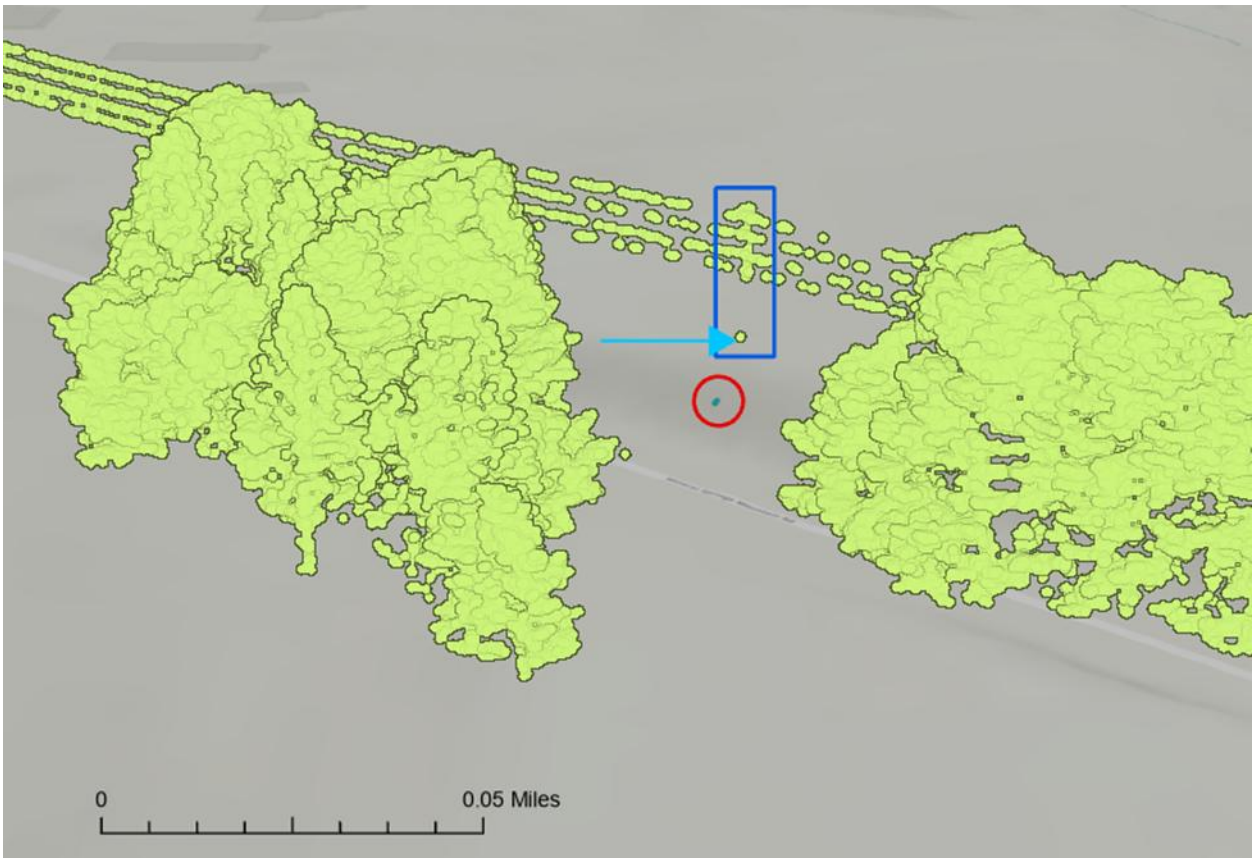


**Fig. 11** A LiDAR-represented utility pole (circled in dark blue) with the closest high vegetation point (light blue arrow) to the geolocated pole location (circled in red)

During validation, we encountered an issue stemming from the use of the LIDAR high vegetation classification to obtain the surrounding context for each geolocated pole. Specifically, a pole clearly visible in both the ArcGIS base map and Google Maps was not captured within the LiDAR high vegetation classification near one of the sampled geolocated locations. As a result, we excluded this pole from the validation set and present validation results for the remaining 29 geolocated poles, as shown in Figure 12.

Validation results indicated that the majority of geolocated poles were accurately positioned relative to their validated locations, with 75% of poles located within 8.4 meters. While six geolocated poles exhibited relatively large geodesic distances (exceeding 10 meters) from their validated locations, overall, the mean geodesic distance was 6.3 meters and the standard deviation was 5.7 meters, with distances ranging from 0.64 to 22.9 meters. More importantly, the perpendicular offset distance from geolocated poles to the nearest road edge was significantly smaller, with 75% of poles located within 2.1 meters and a maximum offset of 7 meters. These results are consistent with the characteristics of our geolocation pipeline, confirming that predicting distance-to-camera is considerably more challenging than predicting a perpendicular offset relative to the road. In addition, the significantly smaller perpendicular offset distance to road edges observed in our validation results is preferable and particularly encouraging, as lateral positioning relative to the roadway is crucial for determining the likelihood of collision, hence a critical factor for road safety and hazard analysis.

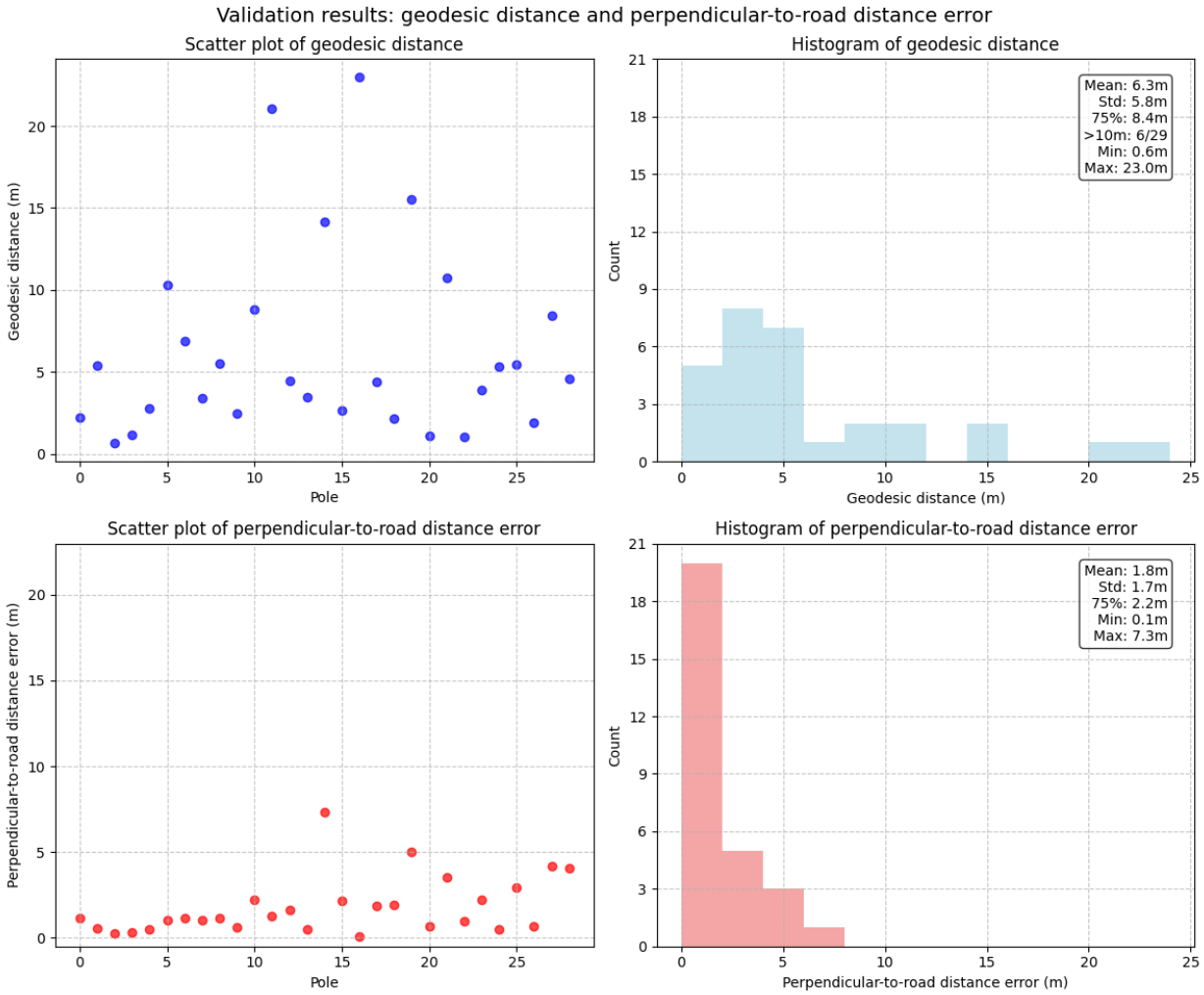


**Fig. 12** Geodesic distances (top) and perpendicular-to-road distance error (bottom) between geolocated and validated pole locations are shown as scatter plots (left) and histograms (right). The perpendicular-to-road distance is more useful from the standpoint of potential hazard detection.

## 4.1 Threshold-Based Precision, Recall, and F1 Analysis

To further characterize geolocation performance, we evaluated precision, recall, and F1 score across a range of geodesic distance thresholds (Table 2 and Figure 13). True positives (TP) were defined as predicted pole locations within the specified geodesic threshold of their validated counterparts, false positives (FP) as predicted poles located beyond the threshold, and false negatives (FN) as validated poles without a corresponding prediction among the 180 predicted locations along the route. The results revealed a stepwise improvement in F1 score, with sharp gains at lower thresholds (increasing from 0.35 at 3 meters to 0.72 at 7 meters), followed by more modest increases before reaching a plateau of 0.93 at 16 meters. This threshold-dependent pattern indicates that, while small thresholds produce limited alignment between geolocated and validated locations, modest increases rapidly improve performance. However, beyond approximately 16 meters, additional threshold increases provide little further benefit. These analyses offer practical guidance for selecting thresholds to achieve reliable geolocation performance.

**Table 2** Precision, recall, and F1 score across geodesic distance thresholds.

| Threshold (m) | TP | FP | FN | Precision | Recall | F1 |
|---|---|---|---|---|---|---|
| 3 | 10 | 19 | 18 | 0.34 | 0.36 | 0.35 |
| 5 | 16 | 13 | 12 | 0.55 | 0.57 | 0.56 |
| 7 | 21 | 8 | 8 | 0.72 | 0.72 | 0.72 |
| 9 | 23 | 6 | 6 | 0.79 | 0.79 | 0.79 |
| 11 | 25 | 4 | 4 | 0.86 | 0.86 | 0.86 |
| 15 | 26 | 3 | 3 | 0.90 | 0.90 | 0.90 |
| 16 | 27 | 2 | 2 | 0.93 | 0.93 | 0.93 |

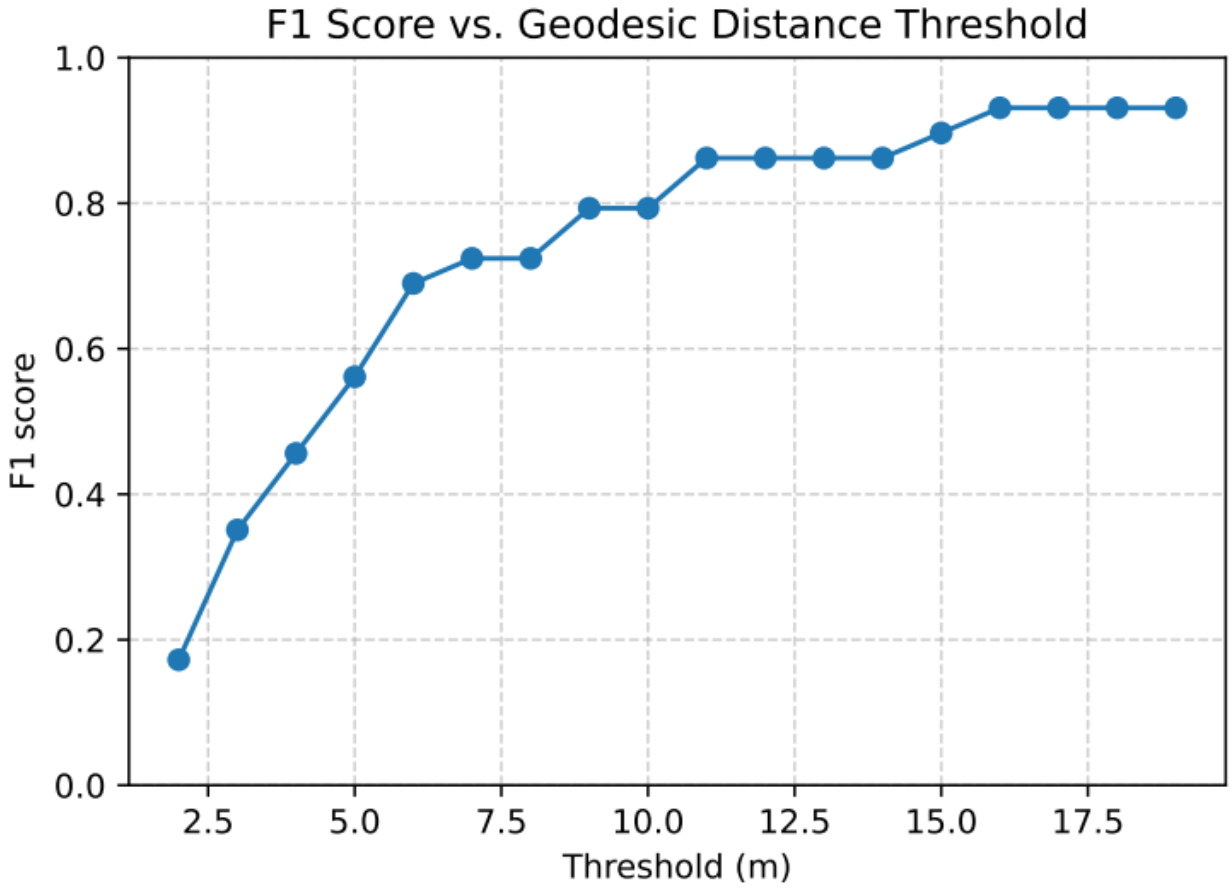


**Fig. 13** F1 score as a function of geodesic distance threshold.

## 4.2 Scenario Classification for Validation Poles

Building on the threshold-based precision, recall, and F1 analysis, we further evaluated the influence of different roadway scenarios on geolocation performance by visually inspecting the surroundings of each validation pole (Table 3). We observed that extraneous features such as driveways and parking lots were strongly associated with larger geodesic distances, particularly for poles with errors greater than 10 meters (e.g., poles 24 and 25, Figure 14). In contrast,

roadway geometry—including straight versus curved segments and the presence of intersections—did not appear to adversely affect accuracy. On the contrary, intersections and curves often provided additional visual landmarks that improved road alignment. Overall, these findings indicate that geolocation accuracy is most sensitive to the quality of road alignment rather than to roadway geometry, and that misclassification of extraneous features as road edges in videolog imagery may degrade alignment and thus increase geolocation error.

**Table 3** Scenario classification for validation poles.

| ID | Geodesic distance | Scenarios | | | |
|---|---|---|---|---|---|
| | | Straight road | Curved road | Intersection | Extraneous features |
| 1 | 0.64 | X | | | |
| 2 | 1.04 | | X | | |
| 3 | 1.09 | X | | X | |
| 4 | 1.16 | X | | X | |
| 5 | 1.89 | X | | | |
| 6 | 2.14 | X | | | |
| 7 | 2.24 | X | | | |
| 8 | 2.49 | X | | | |
| 9 | 2.67 | X | | | |
| 10 | 2.79 | | X | X | |
| 11 | 3.38 | | X | | |
| 12 | 3.44 | X | | | |
| 13 | 3.87 | | X | | |
| 14 | 4.39 | | X | | |
| 15 | 4.46 | X | | X | |
| 16 | 4.60 | | X | | |
| 17 | 5.32 | | X | | |
| 18 | 5.38 | | X | | |
| 19 | 5.47 | X | | X | |
| 20 | 5.54 | | X | X | |
| 21 | 6.91 | X | | | |
| 22 | 8.44 | | X | | |
| 23 | 8.80 | X | | | |
| 24 | 10.32 | | X | | X |
| 25 | 10.74 | X | | | X |
| 26 | 14.18 | X | | | X |
| 27 | 15.53 | | X | | |
| 28 | 21.08 | | X | | |
| 29 | 22.97 | X | | | X |


**Fig. 14**

Fig. 14 Representative validation pole alignment examples (poles 24 and 25). Map overlays (left) show locations of validated poles (pink squares) and geolocated poles (yellow dots) along with camera locations (red dots). LiDAR-image overlays (right) compare projected LiDAR road

edges (blue) with segmented image edges (orange). Extraneous features—a parking lot (pole 24) and a driveway (pole 25)—pulled the LiDAR curved and intersecting road edges toward non-road features, resulting in misalignment and larger geolocation errors.

## 4.3 Performance Metrics and Trade-off Analysis

To assess the practical feasibility of our pipeline for large-scale deployment, we report per-mile runtime, memory usage, and hardware details across major pipeline stages from our HPC cluster run on the test route (approximately 6.3 miles) in Table 4. Stages involving CNN model predictions, such as image segmentation and depth estimation, require GPU acceleration for reasonable performance, while CPU-based stages, such as LiDAR processing and camera parameter estimation via road alignment, support parallel processing that scales with node count. This scalability, enabled by the HPC cluster environment, is essential for deploying the pipeline across all secondary roads (over 54 thousand miles) in the NCDOT videolog. In addition, we analyzed the accuracy/runtime trade-off of increasing levels of image downscaling to compare segmentation model performance. Images in the test route were resized such that the image height was 256, 512, or 1024 pixels and the aspect ratio remained the same as the original image (1200 x 2356 pixels). Due to the cost of resizing the images, downsampling to 1024 pixels did not lead to an appreciable improvement in run time compared to the raw images (3.15 min/mile vs. 3.36 min/mile, respectively). The 512- and 256-pixel downscaling led to run times of 1.28 min/mile and 0.71 min/mile, respectively. However, the time savings were offset by reduced pole observation, with 29.87% (400/1339) of 256-pixel images showing reduced pole instances compared to raw images. 512- and 1024-pixel downscaling returned fewer poles in 12.77% (171/1339) and 5.90% (79/1339) of images, respectively. Based on the 50% time reduction over raw images and the nearly three-fold decrease in pole reduction compared to the 256-pixel images, we selected a 512-pixel image height for our final analysis. A qualitative examination of lane line segmentation yielded similar results, so we applied the same resizing for that component. Finally, we analyzed the trade-offs in our LiDAR road edge detection method, which involves creating a pseudo-image of the route through rasterization of the LiDAR point cloud. We examined the effects of increasing voxel size on rasterization time and outputs, using voxel sizes of 1, 3, 5, and 10 cubic feet. The point cloud for a 100-foot cross-section of the test route contains 34,012,557 coordinates. The run time returns quickly diminish as voxel size increases; highest-hit rasterization and edge detection yielded 5,172,679 points (99,361 edge) in 94 sec (0.25 min/mile) with 1 ft3 voxels, 589,888 (32,528 edge) in 21 sec (0.06 min/mile) at 3 ft3, 217,722 (19,466 edge) in 14 sec (0.04 min/mile) at 5 ft3, and 57,748 (9,665 edge) in 11 sec (0.03 min/mile) at 10 ft3. The decreased point density would lead to better compute efficiency for alignment, but voxel sizes above 1 ft3 led to poor surface mesh reconstruction, resulting in large swaths of visible points being removed during raycasting analysis.

**Table 4** Performance metrics for major pipeline stages from our HPC cluster run with SLURM job scheduler on the test route (approximately 6.3 miles). Runtime is reported in minutes per mile and corresponds to the hardware configuration specified in the table.

| **Pipeline Stage** | **Runtime per Mile (min)** | **Hardware Information** | | |
|---|---|---|---|---|
| | | **# of CPU cores** | **Memory (GB)** | **GPU** |
| Image Segmentation | 1.28 | 16 | 128 | NVIDIA V100S GPU |
| Monocular Depth Prediction | 5.1 | 1 | 64 | NVIDIA V100S GPU |
| LiDAR Processing | 0.25 | 32 | 128 | None |
| Road Lane Detection | 0.09 | 32 | 128 | NVIDIA V100S GPU |
| Camera Parameter Estimation via road alignment | 4.47 | 96 | 256 | None |
| Mapping Input Computation | 1.01 | 64 | 64 | None |
| MRF-Based Object Geolocation | 0.01 | 1 | 64 | None |

# Chapter 5. Conclusions and Future Research

We have presented an approach for detecting and geolocating recurring stationary roadside objects by fusing airborne LiDAR data with videolog images that lack complete camera metadata, building upon the pipeline developed by Krylov et al. [12]. Our work contributes a practical solution to the often-overlooked challenge of sensor fusion with incomplete metadata for roadside asset geolocation and roadway geometry extraction.

Beyond unknown camera parameters, our pipeline also handles the sparsity of images (approximately 26 feet between frames) and a narrow camera field of view (approximately 20 degrees), both of which complicate reliable triangulation using the MRF-based object triangulation approach proposed by Krylov et al. [12]. We found that this triangulation method is particularly sensitive to bearing input errors, and as Krylov et al. [12] noted, it should not be applied for intersections occurring farther than 25 meters from the cameras. Developing geolocation methods that are more robust to bearing inaccuracies remains an important direction for future research. In addition, the precision of initial object segmentation strongly constrains overall geolocation accuracy in our pipeline. Incorporating data fusion directly into the segmentation process could further enhance object detection performance and improve geolocation results.

# Chapter 6. Implementation and Technology Transfer Plan

Regarding the implementation of the outcomes from this project, we have completed image segmentations, image lane segmentations, and depth predictions for all 14 divisions. Currently, we are waiting for the LiDAR data transfer from NCDOT to RENCI before we can start to work on rasterizing LIDAR data, then run our alignment and geolocation pipeline across all divisions. The sheer size of the LIDAR data makes it challenging to transfer LIDAR data for the entire state from NCDOT to RENCI. Due to the complexities of acquiring this data, it is difficult to provide an exact timeline for the final delivery of geolocated pole locations for all 14 divisions at this time. However, we will make every effort to complete the process as soon as possible.

# REFERENCES


[1] S. Y. Alaba and J. E. Ball. "A Survey on Deep-Learning-Based LiDAR 3D Object Detection for Autonomous Driving". In: *Sensors (Basel)* 22.24 (2022), p. 9577. doi: 10.3390/s22249577.

[2] S. F. Bhat et al. *ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth*. 2023. doi: 10.48550/ARXIV.2302.12288.

[3] M. Chaabane et al. "End-to-End Learning Improves Static Object GeoLocalization From Video". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Virtual, 2021, pp. 2063–2072.

[4] H. X. Cheng, X. F. Han, and G. Q. Xiao. "Cenet: Toward Concise and Efficient Lidar Semantic Segmentation for Autonomous Driving". In: *2022 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2022, pp. 01–
06.

[5] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[6] F. Gao and L. Han. "Implementing the Nelder-Mead simplex algorithm with adaptive parameters". In: *Computational Optimization and Applications* 51.1 (2012), pp. 259–277.

[7] C. Godard et al. "Digging into Self-Supervised Monocular Depth Prediction". In: (Oct. 2019).

[8] K. He et al. "Deep residual learning for image recognition". In: *Proc IEEE Conf on CVPR*. 2016, pp. 770–778.

[9] T. Huang et al. "EPNet: Enhancing Point Features with Image Semantics for 3D Object Detection". In: Nov. 2020, pp. 35–52. isbn: 978-3-030-58554-9. doi: 10.1007/978-3-030-58555-6 3.

[10] J. Jain et al. "OneFormer: One Transformer to Rule Universal Image Segmentation". In: 2023.

[11] A. Khan et al. "A survey of the recent architectures of deep convolutional neural networks". In: *Artificial Intelligence Review* 53 (2020), pp. 5455–5516.

[12] V. A. Krylov, E. Kenny, and R. Dahyot. "Automatic discovery and geotagging of objects from street view imagery". In: *Remote Sens.* 10.5 (2018), p. 661.

[13] I. Laina et al. "Deeper Depth Prediction with Fully Convolutional Residual Networks". In: *2016 Fourth International Conference on 3D Vision (3DV)*. 2016, pp. 239–248. doi: 10.1109/3DV.2016.32.

[14] Y. Li et al. "Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.8 (2021), pp. 3412–3432. doi: 10.1109/TNNLS.2020.3015992.

[15] Y. Li et al. " DeepFusion: Lidar-Camera Deep Fusion for Multi-Modal 3D Object Detection ". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2022, pp. 17161–17170. doi: 10.1109/CVPR52688.2022.01667. url: https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01667.

[16] A. Masoumian et al. "Absolute Distance Prediction Based on Deep Learning Object Detection and Monocular Depth Estimation Models". In: *Artificial Intelligence Research and Development*. Oct. 2021. isbn: 9781643682105. doi: 10. 3233/FAIA210151.

[17] R. Mccraith, L. Neumann, and A. Vedaldi. “Calibrating Self-supervised Monocular Depth Estimation”. In: *Proceedings of the Machine Learning for Autonomous Driving Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*. 2020, pp. 1–11.

[18] A.S. Nassar, S. Lef`evre, and J.D. Wegner. “Simultaneous multi-view instance detection with learned geometric soft-constraints”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. Seoul, Korea, 2019, pp. 6559–6568.

[19] A.S. Nassar et al. “GeoGraph: Graph-Based Multi-view Object Detection with Geometric Cues End-to-End”. In: *Proceedings of the European Conference on Computer Vision*. Glasgow, UK, 2020, pp. 488–504.

[20] Gerhard Neuhold et al. “The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[21] R. Ranftl et al. “Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.3 (2022).

[22] E. Shelhamer, J. Long, and T. Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (2017), pp. 640–651. doi: 10.1109/TPAMI.2016. 2572683.

[23] J. Sun et al. *An Empirical Study of Training State-of-the-Art LiDAR Segmentation Models*. 2024. arXiv: 2405.14870 [cs.CV]. url: https://arxiv.org/abs/ 2405.14870.

[24] P. Sun et al. “Object Detection Based on Roadside LiDAR for Cooperative Driving Automation: A Review”. In: *Sensors (Basel)* 22.23 (2022), p. 9316. doi: 10.3390/s22239316.

[25] S. Wang et al. “Object Tracking Based on the Fusion of Roadside LiDAR and Camera Data”. In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–14. doi: 10.1109/TIM.2022.3201938.

[26] Y. Wang et al. “Multi-Modal 3D Object Detection in Autonomous Driving: A Survey”. In: *Int J Comput Vis* 131 (2023), pp. 2122–2152. doi: 10.1007/s11263023-01784-z.

[27] D. Wilson et al. “Object Tracking and Geo-Localization from Street Images”. In: *Remote Sens.* 14.11 (2022), p. 2575. doi: 10.3390/rs14112575.

[28] B. Wu et al. “SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia: IEEE Press, 2018, pp. 1887–1893. doi: 10.1109/ICRA.2018.8462926. url: https://doi.org/10.1109/ICRA.2018.8462926.

[29] D. Wu, Z. Liang, and G. Chen. “Deep learning for LiDAR-only and LiDARfusion 3D perception: a survey”. In: *Intelligence Robotics* 2.2 (2022). issn: 2770–3541. doi: 10.20517/ir.2021.20. url: https://www.oaepublish.com/ articles/ir.2021.20.

[30] E. Xie et al. “SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 12077– 12090. url: https://proceedings.neurips.cc/paper files/paper/2021/file/64f1f27bf1b4ec22924fd0acb550c235-Paper.pdf.

[31] L. Yang et al. “Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data”. In: *CVPR*. 2024.

[32] H. Yi et al. “AI Tool with Active Learning for Detection of Rural Roadside Safety Features”. In: *2021 IEEE International Conference on Big Data (Big Data)*. IEEE. 2021, pp. 5317–5326. doi: 10.1109/BigData52589.2021.9671360.

[33] Fisher Yu et al. “BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[34] W. Zhang et al. “Using Deep Learning to Identify Utility Poles with Crossarms and Estimate Their Locations from Google Street View Images”. In: *Sensors* 18.8 (2018), p. 2484. doi: 10.3390/s18082484.

[35] B. Zhou et al. “Scene Parsing through ADE20K Dataset”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

[36] Bolei Zhou et al. “Semantic understanding of scenes through the ade20k dataset”. In: *International Journal on Computer Vision* (2018).